枢軸選択と丸め誤差 : 高速自動微分法の応用(スーパーコンピュータのための数値計算アルゴリズムの研究)

AUTHOR(S):

久保田, 光一

CITATION:

久保田, 光一. 枢軸選択と丸め誤差 : 高速自動微分法の応用(スーパーコンピュータのための数値計算アルゴリズムの研究). 数理解析研究所講究録 1987, 613: 154-169

ISSUE DATE:

1987-03

URL:

http://hdl.handle.net/2433/99791

数理解析研究所講究録
第613巻 1987年 154-169

154

枢軸選択と丸め誤差———高速自動微分法の応用

東京大学工学部　久保田光一
KOICHI KUBOTA

1．はじめに

高速自動微分法利用のためのプリプロセッサ［久保田，伊理 86］によって，高速自動微分法による関数の高速な勾配計算を実用的に行えるようになった．さらに，計算された値に含まれる丸め誤差の推定も区間解析などで行われていたものより精密に行えるようになった．本稿では線形方程式系の解法を例として取り上げ，数値実験を行うことによって，丸め誤差推定値と実際に発生する丸め誤差との関係を調べ，高速自動微分法による丸め誤差推定の能力を実証する．そして，丸め誤差推定値を基準として，線形方程式系解法における，枢軸選択則やスケーリングの有無の解への影響をみる．また，ここで用いた FORTRAN プリプロセッサ形式の高速自動微分法の利用方法が，スーパーコンピュータを用いることによって計算速度を向上させることができるかどうかについても実験し，報告する．

2．高速自動微分法

高速自動微分法は，多変数関数の勾配を計算する手法であり，次のような特長を持つ．

(1)　関数の正確な勾配の値を高速に計算することができる．すなわち，変数の個数とは無関係に，関数の計算自身に必要な手間の高々定数倍の手間で関数計算と同時に勾配計算を行うことができる．

(2)　勾配の値だけでなく，ヘッセ行列などの高階の導関数の値も正確に求めることができる．

(3)　勾配計算の副産物として，関数の計算値に含まれる丸め誤差の推定値を計算することができる．

一方，従来頻繁に使用されている数値微分では，n 変数関数の勾配を求めるためには $n+1$ 回の関数計算を必要とするので，変数の個数が多くなるほど勾配を求めるために必要な計算量が増加する．さらに，数値微分では計算した勾配の値の有効桁数が減少する“桁落ち”がおこるので，ヘッセ行列などは実用的には

求められなかったものである．また，丸め誤差の推定を実用的に行う手法は今までは無かった．

このように，数値微分法に比べて，(1)，(2)，(3) の特長があることから，高速自動微分法の効力が大きいことは明らかである．

高速自動微分法の詳細は [Iri 1984]，[伊理，土谷，星 1985]，[伊理，久保田 1986] に譲り，ここでは以下で概略を述べるにとどめる．

ある関数が与えられたとして，この関数を計算する過程を考えると，その過程は，個々の演算を頂点とする一種のフローグラフで表される．そのフローグラフの枝に個々の演算によって決まる局所的な偏導関数（“要素的偏導関数”と呼ばれる）を対応付けたものを“計算グラフ”と呼ぶ．この計算グラフを用いると，関数の勾配計算は，“計算グラフ上での最短路計算”とほぼ同様のものとみなすことができる．すなわち，最短路問題の解法が任意の頂点からあるひとつの頂点への最短路のすべてを枝の数に比例する手間で求めるのと同様に，高速自動微分法は関数の計算の途中結果—“中間変数”と呼ぶ—に関する関数の偏導関数のすべてを枝の数に比例する手間で求めるのである．上記の文献中には，関数を計算するために必要な演算回数を $L(f)$ とすれば，関数と勾配とを計算するために必要な演算回数 $L(f, \nabla f)$ は $L(f)$ の4倍以下であることも示されている．

一方，V を関数計算の過程に現れる中間変数の集合，δv を中間変数 v を計算する際の発生丸め誤差とする．すると，関数 f の計算値に含まれる丸め誤差を Δf と記せば，よく知られているように，Δf が小さい範囲では，Δf は

$$\Delta f = \sum_{v \in V} \frac{\partial f}{\partial v} \delta v \tag{2.1}$$

と表すことができる．ここで $|\delta v|$ を $|v|\varepsilon$（ε はいわゆるマシーンε）で評価すれば，Δf の評価 $\overline{\Delta f}$ は

$$\overline{\Delta f} = \sum_{v \in V} \left| \frac{\partial f}{\partial v} v \right| \varepsilon \qquad \text{——（絶対評価）} \tag{2.2}$$

と表現できる．あるいは，δv を $[-|v|\varepsilon, |v|\varepsilon]$ の上の一様分布に従う確率変数とみなして，Δf の評価値として (2.1) の標準偏差

$$\overline{\Delta f} = \left(\frac{1}{3} \sum_{v \in V} \left(\frac{\partial f}{\partial v} v \right)^2 \right)^{1/2} \varepsilon \qquad \text{——（確率評価）} \tag{2.3}$$

を採用することもできる．これらの評価式は今までも考えられないわけではなか

ったが，$\partial f/\partial v$ の計算が困難であったため，実用的ではなかったといえよう．高速自動微分法とそれを手軽に利用するためのシステムによってはじめて，(2.2), (2.3) の評価式を実用的に計算できるようになったといえる．

この高速自動微分法の原理は単純であるが，これを真に実用的なものとするためには，利用のための道具を必要とする．ここで行った実験では FORTRAN プリプロセッサの形で実現されたシステムを利用した．このプリプロセッサは，関数の計算過程を与える FORTRAN の副プログラムを，関数値と同時に勾配の値も計算するような副プログラムに変換するものである．それは，関数の計算過程の中に，IF文，DO文などの制御構造が含まれているようなプログラムも扱うことができるので，複雑な関数の勾配計算も行うことができる．なお，本稿の高速自動微分法の性能評価のための数値実験においては，すべてこのプリプロセッサシステムを用いている．

3．数値実験

線形方程式系の解法について数値実験を行った結果の一部を報告する．

3.1．実験の方針

n 次正方行列 A と n 次元ベクトル $\boldsymbol{b}$ とを与えたときに，$A\boldsymbol{x}=\boldsymbol{b}$ の解 $\boldsymbol{x}$ は A の要素と $\boldsymbol{b}$ の要素とを変数とする関数 $\boldsymbol{x}=\boldsymbol{f}(A,\boldsymbol{b})$ とみなされる．すると，$A\boldsymbol{x}=\boldsymbol{b}$ を解くプログラムは A と $\boldsymbol{b}$ から $\boldsymbol{x}$ を計算する関数の表現とみなされるので，そのプログラムによって計算された解 $\boldsymbol{x}$ に含まれる丸め誤差は高速自動微分法によって推定することができる．

ここでは，ＬＵ分解によって線形方程式系を解く FORTRAN 副プログラム P_1 を作成した．前述のプリプロセッサによって，P_1 を変換してプログラム P_2 を作る．P_2 は高速自動微分法によって関数値，勾配の値，および関数値に含まれる丸め誤差の推定値を計算するプログラムであり，1) 初期化部分，2) 関数計算および計算グラフ作成部分，3) 勾配計算および丸め誤差計算部分　の３個の部分からなる．

この P_2 を実行して，$A\boldsymbol{x}=\boldsymbol{b}$ の解としての関数値 $\boldsymbol{x}$ に含まれる丸め誤差の推定値を得る．以下の実験では，P_1 として枢軸選択則をいろいろに変えたいく

つかのプログラムを取り上げた．そして，同じ関数 $x = f(A, b) = A^{-1}b$ であっても計算手順（プログラム）が異なると計算値に含まれる丸め誤差が異なることを観察した．それにより，どのような計算手順が丸め誤差を小さくするのかを知ることができる．

まず，線形方程式系の解法の入力データとして，次のような“一様乱数行列”を定義する．

定義　一様乱数行列

m 行 n 列の一様乱数行列 $A = (a_{ij})$ とは，a_{ij} が，互いに独立な [-1, 1] の上の一様分布に従うような乱数によって与えられる行列で,階数が $\min(m, n)$ のものである．（この階数に関する条件はほとんど常に満足される．）□

このような一様乱数行列を数値実験の入力データとして採用した理由は，手軽に密行列が作成できることによる．（定義により，以下の実験で用いた一様乱数行列はいずれも正則である.）

次に，実際に発生しうる丸め誤差を実験的に観察するために,“最大丸め誤差”を定義する．

定義　（k-）最大丸め誤差

線形方程式系 $Ax = b$ の行列 A とベクトル b とをあわせて一つの $n(n+1)$ 次元のベクトル a とみなし，$x = f(a)$ と表す．$a^{(0)} = (a_j^{(0)})$ の各要素に次のように最大振れ幅 η をきめてランダムに摂動を与えて k 個の入力データ $a^{(1)}, \ldots, a^{(k)}$ を作る．

$$a_j^{(p)} = a_j^{(0)}(1 + \delta_j^{(p)}),\ 1 \leqq p \leqq k,\ 1 \leqq j \leqq n(n+1),$$

$$\delta_j^{(p)} \ :\ [-\eta, \eta] \text{ の上の一様分布 } (p = 1, \ldots, k).$$

それぞれの入力データ $a^{(i)}$ から単精度で計算された $f(a^{(i)})$ の値を $x^{(i)}$ とし，倍精度で計算して得られる値を丸め誤差のない真の値とみて $\hat{x}^{(i)}$ とする $(i=1, \ldots, k)$.

$x_j^{(i)}$ をベクトル $\boldsymbol{x}^{(i)}$ の第 j 番目の要素として，

$$r_j = \max_{1 \leqq i \leqq k} |x_j^{(i)} - \hat{x}_j^{(i)}|$$

により作られる n 次元ベクトル $\boldsymbol{r} = (r_j)$ を $\boldsymbol{a} = \boldsymbol{a}^{(0)}$ での （k-）最大丸め誤差と定義する．□

もちろん，$\boldsymbol{r}$ の値は $\boldsymbol{a}^{(1)}$,..., $\boldsymbol{a}^{(k)}$ に依存する．また，要素ごとに最大値をとっているので，j が異なれば r_j を与える i（入力データ $\boldsymbol{a}^{(i)}$）も異なりうる．この（k-）最大丸め誤差は，いわば，丸め誤差の実測値といえる．

線形方程式系の解法におけるスケーリングについては，最大要素の大きさを揃えるものと近似解に基づいて行うものとの２種類を取り上げる．それらは以下のように定義される．なお，これらのスケーリングは，後の“逆スケーリング”と区別するために，“順スケーリング”とも呼ぶことにする．

定義　Ｇスケーリング

n 次の正方行列 $A = (a_{ij})$ が与えられたとする．$A\boldsymbol{x} = \boldsymbol{b}$ を

$$\alpha_i = \max_{1 \leqq j \leqq n} \{|a_{ij}|\}, \qquad \beta_j = \frac{1}{\min_{1 \leqq i \leqq n} \{|a_{ij}|\}}$$

から作られる n 次対角行列

$$D_\alpha = \begin{bmatrix} \alpha_1 & 0 & . & 0 \\ 0 & \alpha_2 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \alpha_n \end{bmatrix}, \qquad D_\beta = \begin{bmatrix} \beta_1 & 0 & . & 0 \\ 0 & \beta_2 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \beta_n \end{bmatrix}$$

によって $\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ の形にするスケーリングを“Ｇスケーリング”と呼ぶ．行Ｇスケーリングとは $\tilde{A} = D_\alpha^{-1}A$, $\tilde{\boldsymbol{b}} = D_\alpha^{-1}\boldsymbol{b}$, $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ という変換であり，列Ｇスケーリングは $\tilde{A} = AD_\beta$, $\tilde{\boldsymbol{b}} = \boldsymbol{b}$, $\tilde{\boldsymbol{x}} = D_\beta^{-1}\boldsymbol{x}$ とする変換である．□

定義　Ｓスケーリング [Skeel 1979]

$A\boldsymbol{x} = \boldsymbol{b}$ の近似解を $\tilde{\boldsymbol{x}}$ ($A = (a_{ij})$, $\tilde{\boldsymbol{x}} = (\tilde{x}_j)$) とする．このとき，Ｇスケーリングのときと同様に

$$\alpha_i = \sum_{j=1}^{n} |a_{ij}| \, |\tilde{x}_j| \ , \qquad \beta_j = \tilde{x}_j$$

から作られる n 次対角行列 D_α , D_β によって，$Ax = b$ を $\tilde{A}\tilde{x} = \tilde{b}$ の形にするスケーリングを“Sスケーリング”と呼ぶ．行（列）Sスケーリングは Gスケーリングの場合と同様に定義する．□

次に，上述のスケーリングが解におよぼす影響について調べるために，各行各列の最大要素の大きさがもともと揃う傾向にある一様乱数行列の要素を人為的に不揃いにする“逆スケーリング”を定義する．

定義　べき乗逆スケーリング

任意の定数 ξ に対して，n 次対角行列

$$D_\alpha = \begin{bmatrix} \xi^{-1} & 0 & . & 0 \\ 0 & \xi^{-2} & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \xi^{-n} \end{bmatrix} , \qquad D_\beta = D_\alpha^{-1}$$

によるスケーリングを“べき乗逆スケーリング”と定義する．$Ax = b$ についての行べき乗逆スケーリングは $\tilde{A} = D_\alpha^{-1}A$, $\tilde{b} = D_\alpha^{-1}b$ と変換することであり，列べき乗逆スケーリングは $\tilde{A} = AD_\beta$, $\tilde{x} = D_\beta^{-1}x$ と変換することである．□

定義　乱数逆スケーリング

γ (>0) を決めて，$[0, \gamma]$ の上の一様分布に従う n 個の乱数 $\gamma_1, \ldots, \gamma_n$ を取り出す．n 次対角行列

$$D_\alpha = \begin{bmatrix} e^{-\gamma_1} & 0 & . & 0 \\ 0 & e^{-\gamma_2} & . & 0 \\ . & . & . & . \\ 0 & 0 & . & e^{-\gamma_n} \end{bmatrix} , \qquad D_\beta = D_\alpha^{-1}$$

によるスケーリングを“乱数逆スケーリング”と定義する．行（列）乱数逆スケーリングによる変換は，べき乗逆スケーリングの場合と同様に定義する．□

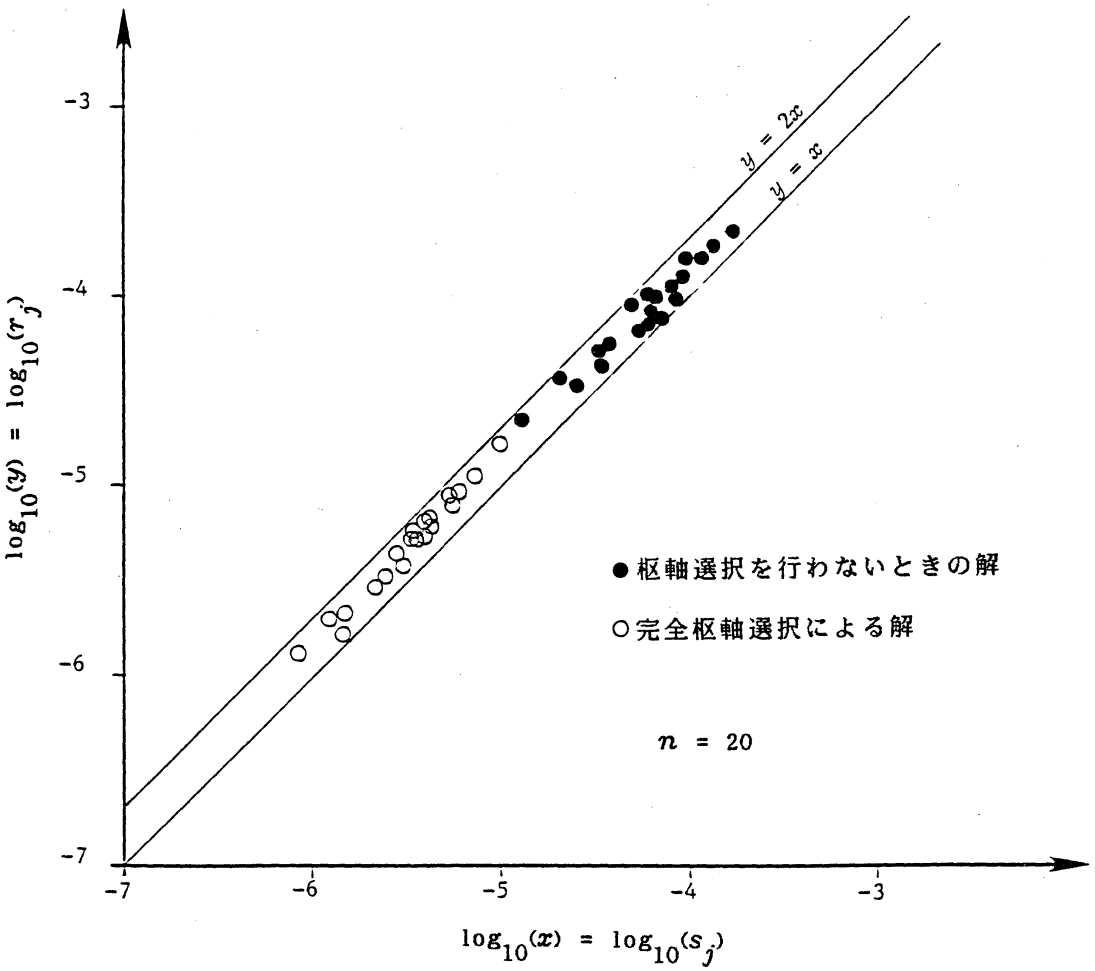

図１．枢軸選択なしの場合と完全枢軸選択を行った場合の解に含まれる最大丸め誤差と丸め誤差評価値

$y = r_j$ ： 最大丸め誤差(n=20, k=10)* ,

$x = s_j$ ： 丸め誤差評価値（確率評価）.

* 20 元線形方程式系を摂動させながら 10 回解いた.

3.2. 数値実験

3.2.1. 実験1. 丸め誤差の実測値と推定値の関係

高速自動微分法により得られるところの計算された値に含まれる丸め誤差の推定値が，その推定値と丸め誤差とを比較することによって，実際に発生する丸め誤差を十分近似していることを実証する．

ここでは，枢軸選択を行わないＬＵ分解と，完全枢軸選択をするＬＵ分解との２種の算法をプログラムにして，これを実験の対象とした．

実験方法は，まず，２０行２０列の一様乱数行列 A と，２０行１列の一様乱数行列（ベクトル）b とを入力データとする．この入力から $k = 10$ のときの（k-）最大丸め誤差（最大振幅 $\eta=10^{-4}$ とする）$r = (r_1,..., r_n)^T$ を求める．次に，プリプロセッサによって変換されたプログラムを実行して，高速自動微分法によって得られる丸め誤差推定値 $s = (s_1,..., s_n)^T$ を求める．最後に，$\log(r_j)$ を縦軸にとり，$\log(s_j)$ を横軸にとって要素ごとに点をプロットする（図１）．

3.2.2. 実験2. 丸め誤差推定値による算法の比較

算法を評価するための基準のひとつとして，計算された値に含まれる丸め誤差の大きさを考える．つまり，算法を実行するプログラムと入力データが与えられれば，高速自動微分法により，その入力データから計算された値に含まれる丸め誤差の値を推定できるので，この推定値を基準としてプログラムの優劣の判断を行うことができる．そこで，このような算法の評価手法の例として，線形方程式系をＬＵ分解によって解く場合に，枢軸選択とスケーリングが計算値に含まれる丸め誤差にどのように影響しているかをみる．

実験は次のように行った．

(1) 入力データとして q 個の (n, n) 一様乱数行列 $A^{(1)},..., A^{(q)}$，$(n, 1)$ 一様乱数行列（ベクトル）$b^{(1)},..., b^{(q)}$ とを作り，グループＡと呼ぶ（最大丸め誤差の場合とは異なり，ひとつのものから摂動して作るものではない）．次に，このグループＡの $A^{(i)}$ と $b^{(i)}$ に対応して，それぞれ行べき乗逆スケーリングしたものを $B^{(1)},..., B^{(q)}$，$c^{(1)},..., c^{(q)}$ とし，グループＢと呼ぶ．さらに，同じ $A^{(i)}$ と $b^{(i)}$ を行乱数逆スケーリングしたものを $C^{(1)},..., C^{(q)}$，$d^{(1)},..., d^{(q)}$ とし，グループＣと呼ぶ．

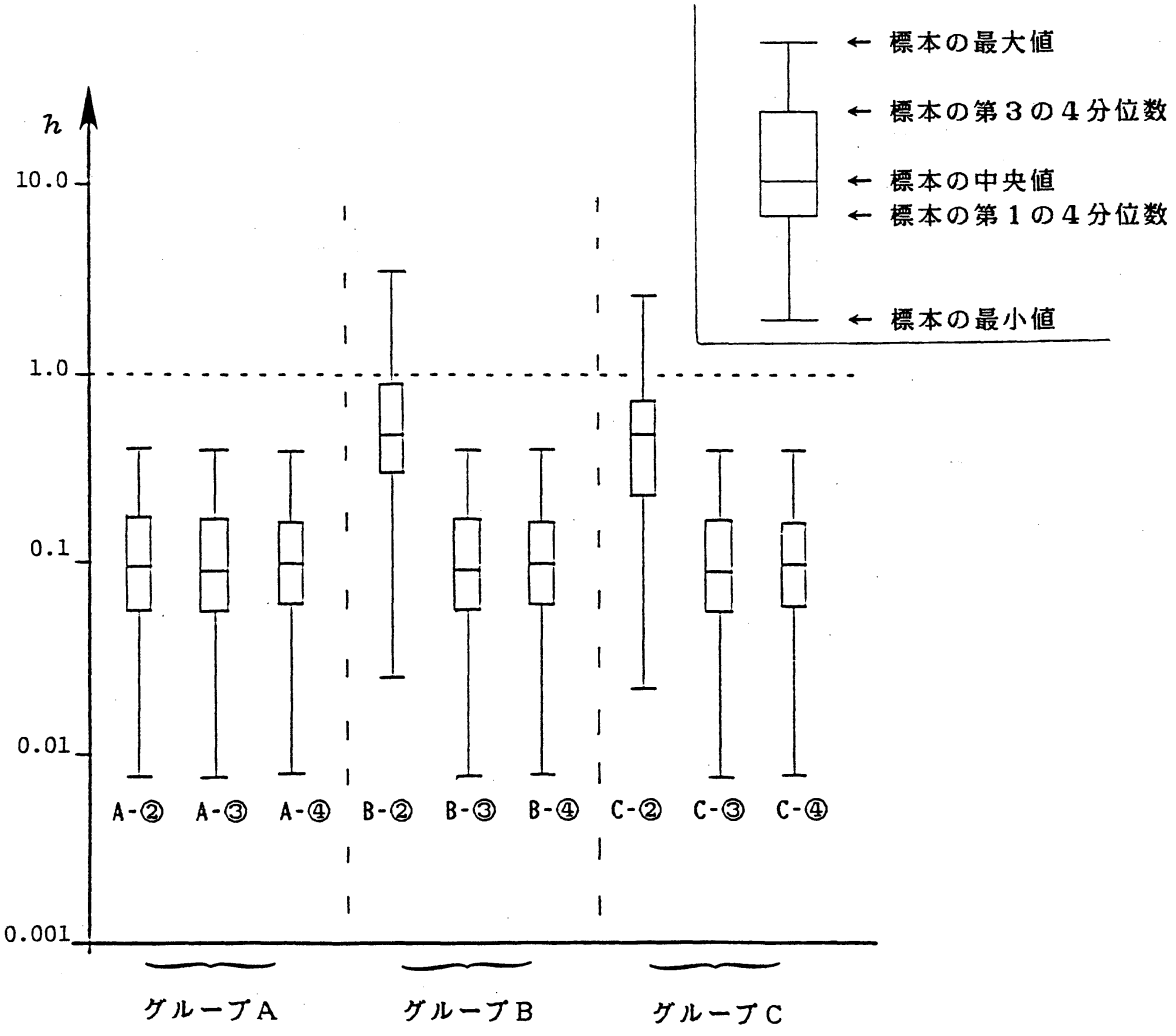

図２．枢軸選択を行わずに解いたときの丸め誤差評価値と枢軸選択を行って解いたときの丸め誤差評価値との比（h）

20 種の 20 元線形方程式系についての実験値．解の要素をすべてまとめているので，一つの標本（サブグループ A-② など）は 400 個の点から成っている．

解法②：行枢軸選択だけを行うＬＵ分解．

③：行Ｇスケーリングの後，行枢軸選択を行うＬＵ分解．

④：行Ｓスケーリングの後，行枢軸選択を行うＬＵ分解．

(2) 次の４通りの規則：

① 枢軸選択もスケーリングも行わない，

② 行の入れ換えを許す枢軸選択（行枢軸選択）だけを行う，

③ 行Ｇスケーリングの後，行枢軸選択を行う，

④ 行Ｓスケーリングの後，行枢軸選択を行う

に従ってＬＵ分解により解を求めるプログラム P_1 を作り，これらをプリプロセッサで P_2 に変換しておく．これらのプログラムに対してそれぞれ入力データグループＡ，Ｂ，Ｃを与えて，計算値に含まれる丸め誤差の推定値を求める．

(3) ①の枢軸選択を行わないＬＵ分解による解は，要素ごとの丸め誤差評価値がＡ，Ｂ，Ｃのどのグループについても同じになるので（詳しくは丸め誤差推定の計算の際に発生する丸め誤差程度の違いはある），これを基準値とする．そして，要素ごとに，Ａ，Ｂ，Ｃの入力データグループに対する ②, ③, ④ の各解法による丸め誤差推定値とその基準値との比 h を調べる．

実際には大きさ (20, 20) の行列を選び，$q = 20$ とした．また，べき乗逆スケーリングの底 ξ としては 5，乱数スケーリングの γ は $10 \cdot \log_e 10$ を選んだ．要素ごとの丸め誤差推定値と①による基準値との比 h をデータグループと解法ごとに求め，図２に表す．また，データグループＢを入力として②で解いたときの結果（図２の B-②）において，それぞれ最大値，中央値，最小値を与える要素を解に持つ３個の行列とベクトルの組を図２から抜き出し，それらの解のすべての要素（２０個ある）について h をプロットする（図３）．

3.2.3．実験３．スーパーコンピュータによる高速化

高速自動微分法によって関数計算の他に勾配と丸め誤差推定値とを計算するプリプロセスされたプログラム（§2のP_2）がスーパーコンピュータを用いることによって高速化される度合を調べる．

まず，線形方程式系を完全枢軸選択則により解くプログラムをプリプロセッサで変換したもの（P_2）を用意する．次に，その P_2 の３個の部分の計算に必要な時間を測定するために，計時機能を呼び出す文を P_2 に挿入する．そして，入力データとして一様乱数行列を用い，行列の次元を 5，10，20 の３種について計算し，その時間を測定する（表１）．

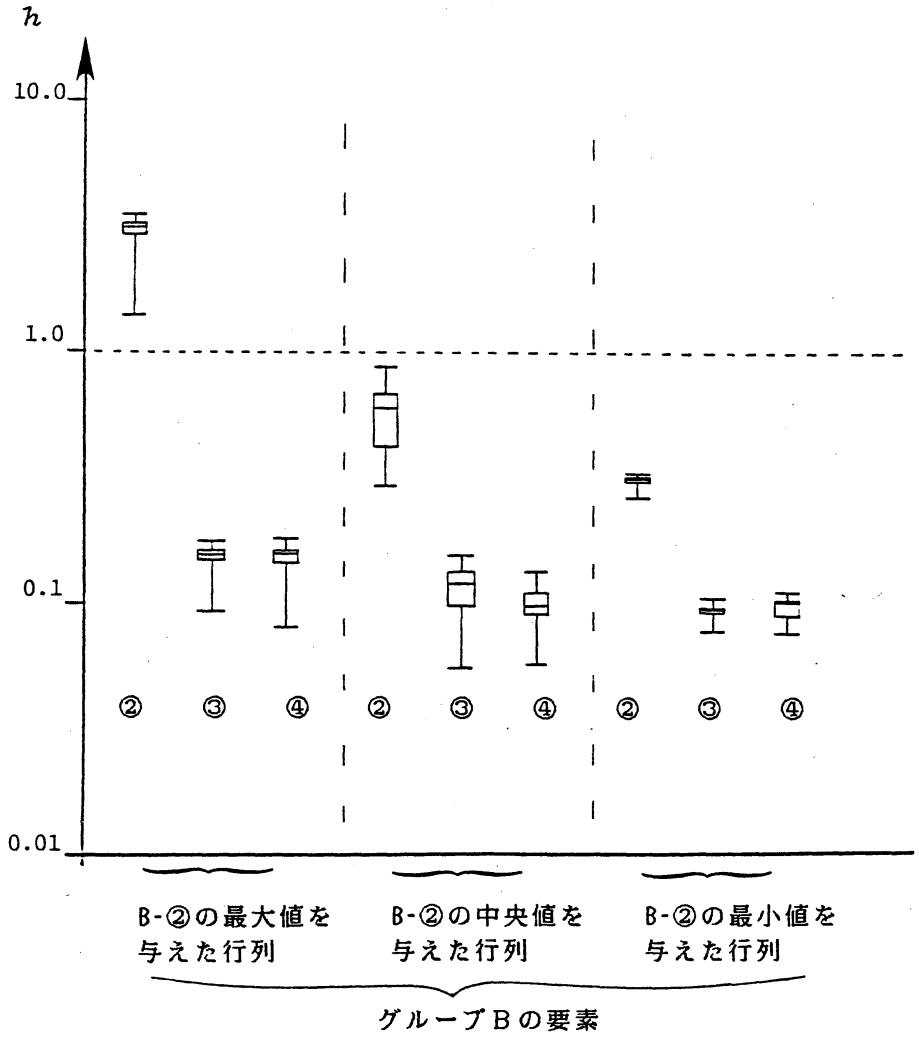

図３．枢軸選択を行わずに解いたときの丸め誤差評価値と枢軸選択を行って解いたときの丸め誤差評価値との比（h）

図２の B-② で最大値（中央値，最小値）を与えた要素を持つ線形方程式系についてその時の他の要素の h を示す．

一つの標本は 20 個（次元数に等しい）の点から成っている．

表1．M680 と S810 での実行時間の比較

単位　1/4800 秒

		M680	S810
SIZE 5 頂点数（191）	T1	1	1
	T2	2	3
変換前　1	T3	1	2
SIZE 10 頂点数（1010）	T1	1	1
	T2	5	11
変換前　3	T3	8	9
SIZE 20 頂点数（6527）	T1	2	2
	T2	34	69
変換前　14	T3	50	59

T1：初期化に要した時間
T2：関数計算と計算グラフ作成に要した時間
T3：勾配計算に要した時間
頂点数：計算グラフの頂点の数
変換前：もとの関数計算プログラムの実行に要した時間
　スカラー演算速度は M680 が 35MIPS，S810 は 19MIPS 相当であるので，ベクトル演算を生かすことのできない高速自動微分法では S810 の方が時間がかかる．

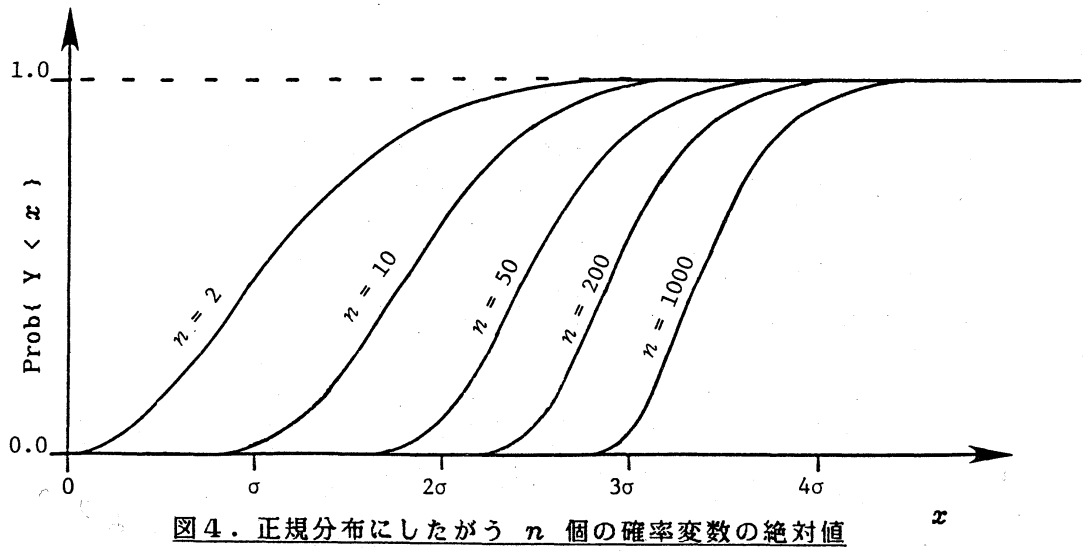

図4．正規分布にしたがう n 個の確率変数の絶対値の最大値 Y の分布

$$Y = \max_{1 \leq i \leq n} \{X_i\}, \quad (X_i \sim N(0, \sigma), \ i = 1, \ldots, n)$$

3.3．考察

3.3.1．丸め誤差推定値と最大丸め誤差との関係

図１では，ほぼ対角線上に点が並んでいて，丸め誤差推定値は，最大丸め誤差の 1.1～1.7 倍の範囲に収まっている．このことから，確率評価 (2.3) による丸め誤差推定値は最大丸め誤差をよく近似していることがわかる．

丸め誤差の分布が平均０分散 σ の正規分布に従うとすれば，大きさ n のサンプルの最大丸め誤差はその正規分布に従う独立な n 個の確率変数の絶対値の最大値の分布（図４）に従う．一方，丸め誤差の確率評価の値は一様分布の和の標準偏差であるが，丸め誤差が正規分布に従うという仮定のもとでは，確率評価の値はその正規分布の標準偏差の近似値であると考えられる．すると，丸め誤差評価値をその正規分布の標準偏差 σ として，最大丸め誤差は図４の $n = 10$ の分布をするはずである．一方，図１からは，$y = x$ と $y = 2x$ の線（図４の σ と 2σ の線に相当する）との間に“最大丸め誤差／確率評価値”の点がほぼ入っていることが読み取れる．このことからも丸め誤差推定値（確率評価）が丸め誤差の標準偏差を与えるというモデルが実際の丸め誤差の発生をよく近似しているとみなすことができよう．

また，図１において，完全枢軸選択を行うと各要素の丸め誤差が約 1/10 に減ることが，最大丸め誤差からもその推定値からも判別できる．したがって，計算方法の違いによる丸め誤差の違いは，丸め誤差推定値だけからでも十分に判別できるといえる．

なお，丸め誤差の分布が正規分布に従うという仮定は，現実的に妥当な仮定である．［土谷 1986］では，より正確に，パラメータによって丸め誤差の分布を一様分布と正規分布の中間的な分布として表している．多くの中間結果が，計算された値の丸め誤差に，関与している場合には，正規分布に似た分布となる．

3.3.2．算法の比較のための丸め誤差評価

まず，図２からわかることは，データグループＡの行列（一様乱数行列のまま）を入力として完全枢軸選択を行って得られた解に含まれる丸め誤差推定値は，枢軸選択を行わない場合のそれに比べて約 1/10 に減少しているということである．さらに，データグループＢ，Ｃの行列を入力した場合には，順スケーリング

を行わずに枢軸選択だけを行って求めた解に含まれる丸め誤差が，枢軸選択を行わないで得た解に含まれるものよりも大きく（悪く）なりうることもわかる（図２の B-②, C-②）．しかし，順スケーリングを行ってから枢軸選択を行えば，丸め誤差評価値の最大値は，枢軸選択を行わないときよりも大きく（悪く）はなっていないことがわかる．すなわち，少なくとも，「枢軸選択を行うならば，あらかじめスケーリングをするべきである」といえよう．

一様乱数行列を用いて，スケーリングを伴わない枢軸選択をした場合には，解のひとつの要素の丸め誤差推定値が大きくなるような行列は，その解の他の要素の丸め誤差推定値も大きくなっている（図３）．そして，丸め誤差評価値の比が要素ごとに極端に違っているような行列とベクトルの組合せはみられなかった．

実験１によって，丸め誤差推定値が実際に発生する丸め誤差の良い限界を与えることがわかっているので，順スケーリングと枢軸選択を行えば，実際に解に含まれる最大丸め誤差は小さくなるであろうことがわかる．

ここで用いたような一様乱数行列では，ＧスケーリングとＳスケーリングとの違いは顕著ではなく，両者の優劣はこの結果からだけでは判断できない．

3.3.3．スーパーコンピュータによる高速自動微分法

ここでは関数計算，勾配計算および丸め誤差推定を高速自動微分法により行うプログラム（§2の P_2）を対象として，スーパーコンピュータの効用を検討する．このプログラムは前述のプリプロセッサによって，FORTRAN の副プログラムの形で表現された関数から，作り出されたものである．したがって，ここで行った実験は我々のプリプロセッサの作り出すプログラムに固有の性能評価である．しかし，汎用計算機（以下 M680（スカラー演算速度 35MIPS））によって高速自動微分法の計算を行った場合と，スーパーコンピュータ（以下 S810（スカラー演算速度 19MIPS程度?））によって行った場合とを対比して眺めてみれば，高速自動微分法と現在のスーパーコンピュータとの適合性を見ることができよう．すなわち，「計算グラフ上で有向道により結ばれていない２個の頂点に対応する演算は同時に実行が可能である」という計算過程に内在する並列性と，現在の計算機との整合性の度合をみることができるはずである．表１から，(1)初期化部分，(2)計算グラフ作成部分 に関しては，計算時間が計算機のスカラー演算速度にほ

ぼ反比例しているようにみえる．(3)勾配計算部分 に関しては，S810 での計算時間の方が M680 での計算時間よりも，スカラー演算速度の比率を考えると，少ないといえる．つまり，勾配計算部分では S810 がいくらか並列性を活かしているようであるが，M680 の方がスカラー演算能力に勝っているので，予想されたことではあるが，現在の利用方法では高速自動微分法に特に S810 を使用することの利点は明らかでない．

4．まとめ

高速自動微分法によって，計算値に含まれる丸め誤差の推定を実用的に行えるようになった．本稿では，このことを実証するための数値実験結果について報告した．特に，丸め誤差推定値を基準として算法の評価を行うという観点から，線形方程式系の解法における枢軸選択とスケーリングが解におよぼす影響を調べた．その結果，丸め誤差推定値は，最大丸め誤差を十分よく近似すること，および，丸め誤差推定値からみて，枢軸選択はスケーリングと共に用いるべきものであることを確かめた．一方，関数の計算過程に内在する並列性を活かすことは，現在我々が作り出しているプログラムと現在のスーパーコンピュータのアーキテクチャでは，まだ困難であることも観察された．

ここで用いた丸め誤差推定のためのモデルが成立する範囲を越えないかぎり，実用上手軽に丸め誤差の推定値が得られるようになったことは意義が大きいと考える．そして，今後の課題は，同様の手法による共役勾配法の性能評価などである．

高速自動微分法全般にわたって御指導を戴いた伊理正夫教授に感謝いたします．

参考文献

W. Baur and V. Strassen (1983): The Complexity of Partial Derivatives. *Theoretical Computer Science*, Vol. 22, pp. 317-330.

M. Iri (1984): Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors — Complexity and Practicality. *Japan Journal of Applied Mathematics*, Vol. 1, No. 2, pp. 223-252.

伊理　正夫，土谷　隆，星　守 (1985): 偏導関数計算と丸め誤差推定の自動化の大規模非線形方程式系への応用．情報処理，Vol. 26, No. 11, pp. 1411-1420.

伊理　正夫，久保田　光一 (1986): 高速微分法とその応用．第７回数理計画シンポジウム論文集，pp. 159-184.

岩田　憲和 (1984): 偏導関数計算の自動化．東京大学大学院工学系研究科情報工学専門課程修士論文．

G. H. Golub and C. F. Van Loan (1983): *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London.

K. V. Kim, Yu. E. Nesterov and B. V. Cherkassky (1985): An Algorithm for Fast Differentiations and Its Applications. *Abstracts of the 12th IFIP Conference on System Modelling and Optimization* (September 2-6, 1985, at Budapest, Hungary), pp. 181-182.

久保田　光一，伊理　正夫 (1986): 高速微分法利用システム— FORTRAN プリプロセッサ．第15回数値解析シンポジウム論文集，pp. 84-87.

V. Yu. Lunin and A. G. Urzhumtsev (1985): Program Construction for Macromolecule Atomic Model Refinement Based on the Fast Fourier Transform and Fast Differentiation Algorithms. *Acta Crystallographica*, Vol. A41, pp. 327-333.

W. Miller and C. Wrathall (1980): *Software for Roundoff Analysis of Matrix Algorithms*. Academic Press, New York.

L. B. Rall (1981): *Automatic Differentiation — Techniques and Applications*. Lecture Notes in Computer Science, Vol. 120, Springer-Verlag, Berlin.

R. D. Skeel (1979): Scaling for Numerical Stability in Gaussian Elimination. *Journal of the Association for Computing Machinery*, Vol. 26, No. 3, pp. 494-526.

土谷　隆 (1986): 高速微分法および丸め誤差推定法とその応用．東京大学大学院工学系研究科計数工学専門課程修士論文．

Yu. M. Volin and G. M. Ostrovskii (1985): Automatic Computation of Derivatives with the Use of the Multilevel Differentiating Technique - 1. Algorithmic Basis. *Computers and Mathematics with Applications*, Vol. 11, No. 11, pp. 1099-1114.